Moderately Hard Functions: From Complexity to Spam Fighting

Moni Naor[1] *

Weizmann Institute of Science
Rehovot 76100, Israel
naor@wisdom.weizmann.ac.il

Abstract. A key idea in cryptography is using hard functions in order to obtain secure schemes. The theory of hard functions (e.g. one-way functions) has been a great success story, and the community has developed a fairly strong understanding of what types of cryptographic primitives can be achieved under which assumption.
We explore the idea of using *moderately* hard functions in order to achieve many tasks for which a perfect solution is impossible, for instance, denial-of-service. We survey some of the applications of such functions and in particular describe the properties moderately hard functions need for fighting unsolicited electronic mail. We suggest several research directions and (re)call for the development of a theory of such functions.

1 Introduction

Cryptography deals with methods for protecting the privacy, integrity and functionality of computer and communication systems. One of the key ideas of Cryptography is using intractable problems, i.e. problems that cannot be solved effectively by any feasible machine, in order to construct secure protocols. There are very tight connections between Complexity Theory and Cryptography (which was called “a match made in heaven” [34]). Over the last 20 years the theory of hard functions (e.g. one-way functions) has been a great success story and we have a pretty good understanding of which tasks require which computational assumptions (see Goldreich’s book(s) [31] for a recent survey).

However, as we will see later, many tasks, ranging from very concrete ones such as combatting spam mail to fairly abstract such as few-round zero-knowledge require a finer notion of intractability, which we call *moderate hardness*. In general one can say that while there are many applications where moderate intractability is needed, the study of moderate hardness has been neglected, compared with strict intractability.

Today, the Internet makes cryptographic protection more necessary than ever. It also provides better opportunities than ever for performing distributed tasks. But many of these applications are either impossible without the use of moderately hard functions

* Incumbent of the Judith Kleeman Professorial Chair. Research supported in part by a grant from the Israel Science Foundation.

(e.g. fair contract signing without trusted third parties [12], see below) or prone to abuse (various denial of service attacks, free-loading etc.).

We survey some of the applications where moderate hardness was used in an essential way in Section 3, starting with an emphasis on spam fighting in Section 2. In Section 4 we outline future research direction in the area.

2 Pricing via Processing or Combatting Junk Mail

Unsolicited commercial e-mail, or spam, is a problem that requires no introduction, as most of us are reminded of it daily. It is more than just an annoyance: it incurs huge infrastructure costs (storage, network connectivity etc.). There are several (not mutually exclusive) approaches for combatting spam mail, but the one that interests us is the computational approach to fighting spam, and, more generally, to combatting denial of service attacks. This approach was initiated by Dwork and Naor [24]:

> "If I don't know you and you want to send me a message, then you must prove you spent, say, ten seconds of CPU time, just for me and just for this message."

The "proof of effort" is cryptographic in flavor; as explained below, it is a moderately hard to compute (but very easy to check) function of the message, the recipient, and a few other parameters. The system would work automatically and in the background, so that the typical user experience is unchanged. Note that filtering and computational spam fighting are complementary, and the techniques reinforce each other.

The Computational Approach: In order to send a message m, the sender is required to compute a tag

$$z = f(m, sender, receiver, time)$$

for a moderately hard to compute "pricing" function f. The message m is transmitted together with z, the result of the computation. Software operating on behalf of the receiver checks that z is properly computed *prior to* making m visible to the receiver. It is helpful to think of the tag z as a *proof of computational effort*. The function f is chosen so that:

1. The function f is not amenable to amortization or mass production; in particular, computing the value $f(m, sender, \mathsf{Alice}, time)$ does not help in computing the value $f(m, sender, \mathsf{Bob}, time)$. This is key to fighting spam: the function must be recomputed for each recipient (and for any other change of parameters, including time).
2. There is a "hardness" parameter to vary the cost of computing f, allowing it to grow as necessary to accommodate the increased computing power, as predicted by Moore's Law.
3. There is an important difference in the costs of computing f and of checking f: the cost of sending a message should grow much more quickly as a function of the hardness parameter than the cost of checking that a tag has been correctly computed. For the functions described in [24] and [5], the cost of computing the tag grows exponentially in the hardness parameter; the cost of checking a tag grows

only linearly. This difference allows the technique to accommodate great increases in sender processor speeds, with negligible additional burden on the receiver.

We want the cost of checking a computation to be very low so that the ability to wage a denial of service attack against a receiver is essentially unchanged. In other words, we don't want the spam-fighting tools to aid in denial of service attacks. In fact, the pricing via processing approach has been suggested also as a mechanism for preventing distributed denial of service attacks [36] and for metering access to web pages [30]

Dwork and Naor [24] called such a function a *pricing function* because the proposal is fundamentally an economic one: machines that currently send hundreds of thousands of spam messages each day, could, at the 10-second price, send only eight thousand.

2.1 Memory-bound Functions

The pricing functions proposed in [24] and other works such as [5] are all CPU-bound. The CPU-bound approach might suffer from a possible mismatch in processing among different types of machines (PDAs versus servers), and in particular between old machines and the (presumed new, top of the line) machines that could be used by a high-tech spam service. A variant on the pricing via processing approach outlined above is to exploit the fact that memory with small latency (caches) is small in computers. In order to address these disparities, Abadi et al [1] proposed an alternative computational approach based on memory latency: design a pricing function requiring a moderately large number of scattered memory accesses. The ratios of memory latencies of machines built in the last five years is less than four; a comparable number for desktop CPU speeds is over 10. The CPU speed gap is even greater if one includes low-power devices as PDAs, which are also used for e-mail. Thus, functions whose cost is principally due to memory latency may be more equitable as pricing functions than CPU-bound functions.

This raises an interesting question: which functions require many random accesses to a large memory? More specifically, can we suggest functions satisfying the above requirements from a pricing function and where the memory access dominates CPU work for all "reasonable" settings. Dwork, Goldberg and Naor [23] have explored this direction and proposed functions based on random walks in a large shared random table. For an abstract version of their proposal they managed to give a tight lower bound on the amortized number of memory access when computing an acceptable proof of effort, based on idealized hash functions (random oracles). They suggested a very efficient concrete implementation of the abstract function, inspired by the RC4 stream cipher.

In addition to their application for spam fighting, the memory-bound functions of [23] have been proposed to prevent Sybil Attacks [44], where an adversary pretends to be many different processors (which is very relevant in peer-to-peer applications [21]).

3 Other Applications of Moderate Hardness

We now list a few other areas where applying moderately hard functions proved useful. As one can see the range is quite large.

Time-Locks: Is it possible to "ice-lock" a piece of information so that it will only be available in the far future but not in the near future? This is one of the more intuitive applications of moderately hard functions, i.e. the data is encrypted with a weak key so that finding it must take time. Several researchers have considered this case, since the early days of Cryptography. However most of them ignored the issue of parallel attacks, i.e. simply taking a DES key and revealing a few bits is not a real time-lock since the adversary may utilize a parallel machine for the exhaustive search, thus speeding up the lock significantly. The first scheme taking into account the parallel power of the attacker is the one by Rivest, Shamir and Wagner [42]. They suggested using the "power function", i.e. computing $f(x) = g^{2^{2^x}} \pmod N$ where N is a product of two large primes[1]. Without knowing the factorization of N the best that is known is repeated squaring - a very sequential computation in nature. The scheme has been used to create a time capsule called LCS35 at MIT [41]. In general the power function is very useful and was used to create many other construction (see below).

Timed Commitments: A string commitment protocol allows a sender to commit, to a receiver, to some value. The protocol has two phases. At the end of the *commit* phase the receiver has gained no information about the committed value, while after the *reveal* phase the receiver is assured that the revealed value is indeed the one to which the sender originally committed. Timed commitments, defined and constructed by Boneh and Naor [12], are an extension of the standard notion of commitments in which there is a potential forced opening phase permitting the receiver, by computation of some moderately hard function, to recover the committed value without the help of the committer. Furthermore, the future recoverability of the committed value is verifiable: if the commit phase ends successfully, then the receiver is correctly convinced that forced opening will yield the value. In [12] such commitments based on the power function were constructed.

Fair Contract Signing: An important application as well as motivation for the timed-commitment of [12] was fair contract signing: two mutually suspicious parties wish to exchange signatures on a contract. How can one party assure that once it has provided its signature on the contract it will also receive the other parties signature (on the same or other message). The problem of fair contract signing was the impetus of much of the early research on cryptographic protocols. Protocols where no third party (judge, referee) is involved are based on the *gradual release of signatures/secrets*. Examples include [10, 28, 17, 20, 38, 12]. In such cases it can be shown that timing considerations are essential for fairness.

Collective Coin Flipping: Another application of timed commitments is to the problem of collective coin flipping. Suppose that two parties A and B wish to flip a coin in such a matter that (i) the value of the coin is unbiased and well defined even if one of the parties does not follow the protocol (if both of them don't follow, then it is a lost case) (ii) if both parties follow the protocol, then they agree on the same value for the coin. The results of Cleve and Impagliazzo [16, 18] imply that with out timing consideration it is impossible to achieve this goal. One the other hand,

[1] This is related to the random access property of the Blum-Blum-Shub generator [11].

in [12] timed commitments are used to construct such protocols, by making weak assumption on the clocks of the two parties.

Key Escrow: Suppose that a central authority wishes to escrow the keys of the users in order to facilitate wiretapping. This is of course very controversial and one would like to limit the power this central authority has. Bellare and Goldwasser [7, 8] suggested "time capsules" for key escrowing in order to deter widespread wiretapping. A major issue there is to verify at escrow-time that the right key is escrowed. Similar issues arise in the timed commitment work, where the receiver should make sure that at the end of the commit phase the value is recoverable. Other applications include Goldschlag and Stubblebine [33] and Syverson [45].

Benchmarking: Cai et al [13] were interested in coming up with *uncheatable benchmarks*, i.e. methods that allow a system owner to demonstrate the computational power of its system. They suggested using the power function for these purposes. Note that here no measures are taken or needed in order to ensure the correctness of the values.

Zero-knowledge Protocols: Using moderately hard functions enables us to construct zero-knowledge protocols in various settings where either it is not known how to do it without timing considerations or even provably impossible. This includes concurrent zero-knowledge [26, 37, 43, 15], Resettable Zero-Knowledge [14, 25] and three-round zero-knowledge [32, 25]. Furthermore Dwork and Stockmeyer [27] investigated the possibility of constructing nontrivial zero-knowledge interactive proofs under the assumption that the prover is computationally bounded during the execution of the protocol.

4 Open Problems in Moderately Hard Functions

We view the focus on moderate hardness as an adjustment of the theoretical models of computation to the constraints – and the benefits – of the real world. In the real world, we have clocks, processors have cycle speeds and memory latencies behave in certain ways. Why should the theory ignore them? We would like to see a comprehensive theory of moderately hard functions. Ideally, given such a theory we would be able to characterize and know for most tasks (both those described above and unforseen ones) the assumptions needed, and offer good schemes to obtain them.

There are many interesting research questions that are largely unexplored in this area. Some of the questions mirror the hard functions world and some are unique to this area. Also some questions are relevant to all or most applications described above, whereas others are more specialized. We now propose several research directions. This list is the outcome of joint work with Dan Boneh and Cynthia Dwork.

Unifying Assumption: In the intractable world we have the notion of a one-way function which is both necessary for almost any interesting task[2] and sufficient for many of them. Is there such a unifying assumption in the moderately hard world? Alternatively, is there evidence for a lack of such assumption (e.g. in the style of oracle separation, a la Impagliazzo and Rudich [35]).

[2] Threshold secret sharing and encryption using a very long shared key (one-time pad) are an exception.

Precise Modelling: Unlike the tractable vs. Intractable case the exact machine model (Turing Machines, RAMS) is not very significant, here given that exact time estimate may be important it may matter. For each application we have in mind we have to describe the exact power of the adversary as well as a model of computation. Such modelling was carried out in the work on memory-bound functions [23]. Given the diversity of applications of moderately hard functions the issue of a unifying model comes up.

Hardness Amplification: Suppose that we have a problem that is somewhat hard and we would like to come up with one that is harder. How should we proceed. For example, can we amplify a problem by iterating it on itself?

Hardness vs. Randomness: A question that is relevant to some applications is the connection between (moderate) hardness and (moderate) pseudorandomness. This is particularly important for applications such as timed commitment, where the protection to the data has to ensure (temporary) indistinguishability from randomness. Note that following the existing reductions of the intractable world, going from one-way functions (or, for simplicity, permutations) to pseudo-random generators, is too heavy given the iterative nature of the construction.

Evidence for non-amortization: Suppose that there is an adversary that attempts to solve many problems simultaneously and manages to obtain a *marginal* work factor which is smaller than that of individual instances. For some applications, such as timed commitment, this is not significant, but for the pricing-via-processing approach this is devastating. Therefore the question is what evidence do we have for the infeasibility of mass production. For instance, is it possible to demonstrate that if a certain problem is not resilient to amortization, then a single instance of it can be solved much more quickly?

The possibility of assumption free memory-bound functions: For the memory-bound functions an intriguing possibility is to apply recent results from complexity theory in order to be able to make unconditional (not relying on any assumptions) statements about proposed schemes. One of the more promising directions in recent years is the work on lower bounds for branching program and the RAM model by Ajtai [3, 4] and Beame et al [6]. It is not clear how to directly apply such results.

Immunity to Parallel Attacks For timed commitment and time-locking, it is important to come up with moderately hard functions that are immune to parallel attacks, i.e., solving the challenge when there are many processors available takes essentially the same time as when a single one is at hand. In [42, 12] and other works the power function was used, but there is no good argument to show immunity against parallel attacks. An intriguing possibility is to reduce worst-case to average case, i.e., find a random self reduction. While in the intractable world it is known that there are many limitations on random self reductions (see [29]), in this setting it is not clear that one cannot use them to demonstrate average-case hardness. In particular, is it possible to randomly reduce a P-Complete problem to itself. Such a reduction would yield a distribution on problems that at least has *some* inherent sequentiality. More specifically is it possible to use linear programming or lattice basis reduction for such purposes?

New Candidates for Moderately Hard Functions We need more proposals for moderately hard functions with various conjectured (or provable) properties, for instance on the amount of memory required to compute them. Areas such as Algebraic Geometry, Lattices and Linear Programming are promising.

Acknowledgements

Most of my work on the topic of moderately hard functions was carried out with Dan Boneh and Cynthia Dwork. Furthermore, the discussion above on future directions of research is the result of joint work with Cynthia and Dan. I thank them for many years of fruitful collaboration.

References

1. M. Abadi, M. Burrows, M. Manasse, and T. Wobber, *Moderately Hard, Memory-Bound Functions*, Proceedings of the 10th Annual Network and Distributed System Security Symposium February, 2003.
2. M. Ajtai, *Generating Hard Instances of Lattice Problems*, 28th Annual Symposium on Theory Of Computing (STOC), 1996, pp. 99–108.
3. M. Ajtai, *Determinism versus Non-Determinism for Linear Time RAMs*, STOC 1999, pp. 632–641.
4. M. Ajtai, *A Non-linear Time Lower Bound for Boolean Branching Programs*, FOCS 1999, pp. 60–70.
5. A. Back, Hashcash - A Denial of Servic Counter-Measure, available at `http://www.cypherspace.org/hashcash/hashcash.pdf`.
6. Paul Beame, Michael E. Saks, Xiaodong Sun, Erik Vee, *Super-linear time-space tradeoff lower bounds for randomized computation,* FOCS 2000, pp. 169–179.
7. M. Bellare and S. Goldwasser, *Verifiable Partial Key Escrow*, Proc. of 4th ACM Symp. on Computer and Communications Security, 1997, pp. 78–91.
8. M. Bellare and S. Goldwasser, *Encapsulated key escrow*. MIT Laboratory for Computer Science Technical Report 688, April 1996.
9. M. Ben-Or, O. Goldreich, S. Micali and R. L. Rivest, *A Fair Protocol for Signing Contracts*, IEEE Transactions on Information Theory 36/1 (1990) 40-46
10. M. Blum, *How to Exchange (Secret) Keys*, Proc. 15th ACM Symp. on Theory of Computing, 1983, pp. 440–447 and ACM TOCS 1(2): 175-193 (1983)
11. L. Blum, M. Blum and M. Shub, *A Simple Unpredictable Pseudo-Random Number Generator*, SIAM J. Comput. 15(2): 364-383 (1986).
12. D. Boneh and M. Naor, *Timed Commitments*, Advances in Cryptology - CRYPTO'2000 Proceedings, Lecture Notes in Computer Science No. 1880, Springer, 2000, pp. 236–254.
13. J. Y. Cai, R. J. Lipton, R. Sedgwick, A. C. Yao, *Towards uncheatable benchmarks, Structures in Complexity*, Proc. Structures in Complexity, pp. 2–11, 1993.
14. R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. *Resettable Zero-Knowledge*, ECCC Report TR99-042, Oct 27, 1999 and Proc. of 32nd ACM Symp. on Theory of Computing, 2000, pp. 235–244.
15. R. Canetti, J. Kilian, E. Petrank and A. Rosen. *Concurrent Zero-Knowledge Requires* $\tilde{\Omega}(\log n)$ *Rounds*, Proc. of the 33rd ACM Symposium on the Theory of Computing, 2001, pp. 570–579. Full version: Electronic Colloquium on Computational Complexity, Report TR01-050, Volume 8, 2001. Available: `www.eccc.uni-trier.de/eccc/`
16. R. Cleve, *Limits on the Security of Coin Flips when Half the Processors Are Faulty*, Proc. of 18th ACM Symp. on Theory of Computing, 1986, pp. 364–369.

17. R. Cleve, *Controlled gradual disclosure schemes for random bits and their applications*, Advances in Cryptology – Crypto'89, Lecture Notes in Computer Science vol. 435, Springer, 1990, pp. 573–588.
18. R. Cleve, R. Impagliazzo, *Martingales, collective coin flipping and discrete control processes*, manuscript, 1993.
Available: `http://www.cpsc.ucalgary.ca/~cleve/papers.html`
19. I. Damgård, *Concurrent Zero-Knowledge in the auxiliary string model*, Advances in Cryptology – EUROCRYPT '2000, Lecture Notes in Computer Science vol. 1807, Springer, 2000, pp. 418–430. .
20. I. Damgård, *Practical and Provably Secure Release of a Secret and Exchange of Signatures*, J. of Cryptology 8(4): 201-222 (1995)
21. J. Douceur, *The Sybil Attack*, Proc. IPTPS 2002.
22. D. Dolev, C. Dwork and M. Naor, *Non-malleable Cryptography*, Siam J. on Computing, vol. 30(2), 2000, pp. 391–437.
23. C. Dwork, A. Goldberg and M. Naor, *On Memory-Bound Functions for Fighting Spam*, Advances in Cryptology – CRYPTO 2003 Proceeding, Lecture Notes in Computer Science, Springer, 2003.
24. C. Dwork and M. Naor, *Pricing via Processing -or- Combatting Junk Mail*, Advances in Cryptology – CRYPTO'92, Lecture Notes in Computer Science vol. 740, Springer, 1993, pp. 139–147.
25. C. Dwork and M. Naor, *Zaps and their applications*, Proc. 41st IEEE Symp. on Foundations of Computer Science, 2000, pp. 283–293. Also: Electronic Colloquium on Computational Complexity (ECCC)(001): (2002).
26. C. Dwork, M. Naor and A. Sahai, *Concurrent Zero-Knowledge*. Proc. of the 30th ACM Symposium on the Theory of Computing, 1998, pp. 409–418.
27. C. Dwork and L. Stockmeyer, *2-Round Zero-Knowledge and Proof Auditors*. Proc. of the 34th ACM Symposium on Theory of Computing (2002), pp. 322–331.
28. S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, CACM 28(6): 637-647 (1985).
29. J. Feigenbaum, L. Fortnow, *Random-Self-Reducibility of Complete Sets*, SIAM J. Comput. 22(5): 994-1005 (1993)
30. M. Franklin and D. Malkhi, Auditable metering with lightweight security. Journal of Computer Security, Vol. 6, No. 4, 1998
31. O. Goldreich. *Foundation of Cryptography – Basic Tools*. Cambridge University Press, 2001.
32. O. Goldreich and H. Krawczyk, *On the Composition of Zero Knowledge Proof Systems*, SIAM J. on Computing, Vol. 25, No. 1, pp. 169–192, 1996.
33. D. Goldschlag and S. Stubblebine, *Publicly Verifiable Lotteries: Applications of Delaying Functions*, Financial Cryptography, Second International Conference, Lecture Notes in Computer Science vol. 1465, Springer, 1998, pp. 214–226.
34. S. Goldwasser, *New directions in cryptography: Twenty some years later,* Proceedings of 38th Annual Symposium on Foundations of Computer Science, IEEE, 1997, pp. 314–324.
35. R. Impagliazzo and S. Rudich, *Limits on the provable consequences of one-way permutations*, STOC 1989, pp. 44–61.
36. Ari Juels, John Brainard, *Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks*.
37. J. Kilian, E. Petrank and C. Rackoff, *Lower Bounds for Zero Knowledge on the Internet*, IEEE 38th Symp. on Foundations of Computer Science, 1998, pp. 484–492.
38. M. Luby, S. Micali and C. Rackoff, *How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin*, Proc. IEEE Symp. Foundations of Computer Science, 1983, pp. 11–21

39. A. Menezes, P. van Oorschot and S. Vanstone, **Handbook of Applied Cryptography**, CRC Press, 1996.
40. M. Naor, B. Pinkas and R. Sumner, *Privacy Preserving Auctions and Mechanism Design*, Proc. of the 1st ACM conference on E-Commerce, November 1999, pp. 129 – 139.
41. R. Rivest, *Description of the LCS35 Time Capsule Crypto-Puzzle*, April 4, 1999, available: http://www.lcs.mit.edu/research/demos/cryptopuzzle0499
42. R. Rivest, A. Shamir and D. Wagner, *Time lock puzzles and timed release cryptography*, Technical report, MIT/LCS/TR-684
43. A. Rosen, *A note on the round-complexity of concurrent zero-knowledge*, Advances in Cryptology - CRYPTO'2000 Proceedings, Lecture Notes in Computer Science Vol. 1880, Springer, 2000, pp. 451–468.
44. D. H. S. Rosenthal, M. Roussopoulos, P. Maniatis and M. Baker, *Economic Measures to Resist Attacks on a Peer-to-Peer Network*, Proceedings of the Workshop on Economics of Peer-to-Peer Systems, June 2003.
45. P. Syverson, ”Weakly Secret Bit Commitment: Applications to Lotteries and Fair Exchange”, Proceedings of the 1998 IEEE Computer Security Foundations Workshop (CSFW11), June 1998.